Abstract

Admissible and consistent heuristic functions are usually preferred in single-agent heuristic search as they guarantee optimal solutions with complete search methods such as A* and IDA*. Real-time problems, however, frequently make a complete search intractable due to space and/or time limitations. For instance, a path-planning agent in a real-time strategy game may need to take an action before its complete search has the time to finish. In such cases, incomplete search techniques (such as RTA*, SRTA*, RTDP, DTA*) can be used. Such algorithms conduct a limited ply lookahead and then evaluate the states envisioned using a heuristic function. The action selected on the basis of such evaluations can be suboptimal due to the incompleteness of search and inaccuracies in the heuristic. It is usually believed that deeper lookahead increases the chances of taking the optimal action. We demonstrate that it is not necessarily the case and that selecting lookahead depth dynamically can significantly improve the performance.

1 Real-time Heuristic Search

The basic framework of real-time heuristic search is as follows. The agent traverses a state space by taking an action in each state. Its goal is to reach one of the predetermined “goal” states. The standard assumption is that the state space, the action set, and the set of goal states are fixed. Thus, each problem instance can be described by the agent’s initial state. Note that such search framework can be easily extended to more general decision-making. One way of doing so is via Markov Decision Processes as it is usually done in the field of Reinforcement Learning [1].

Throughout the paper, we will assume that the agent has a perfect domain model of the environment but cannot always tell which states are better than others. Thus, it is forced to use a heuristic estimate (henceforth heuristic function) of the state quality or value. Often such heuristic function is an estimate on the distance between the state in question and the closest goal state.

Complete search methods such as A* [8] and IDA* [10] produce optimal solutions when based on an admissible heuristic function. The primary drawbacks are the exponential running time and the necessity to wait until the search completes before the first action can be taken [11]. This limits the applicability of complete search in practice as the deliberation time per action can be severely limited [9], the domain model can be expensive, the goal states can be difficult to recognize [13]. Consequently, despite numerous advances in improving heuristic functions [12, 21], incomplete real-time/on-line search methods remain the practical choice for complex real-life problems.

Various real-time search methods have been proposed including: RTA* [11], RTDP [1], SRTA* and DTA* [22]. Such algorithms base their decisions on heuristic information collected from a partial tree expansion (lookahead) prior to reaching the goal state. Since the heuristic function is generally inaccurate and the search is incomplete, suboptimal solutions can be produced even with admissible and consistent heuristics.

It is widely believed that looking ahead deeper improves the solution quality (e.g., [11]). Consequently, a considerable amount of effort has been put into increasing the lookahead depth by using selective search (search extensions) and hardware/software optimizations.

In this paper we demonstrate that looking ahead deeper can actually decrease the chances of taking the optimal action as well as the overall solution quality. This phenomenon is known as lookahead pathology. Additionally, we show that selecting the lookahead depth dynamically can greatly improve the solution quality.

2 Related Work & Our Novel Contributions

Lookahead pathologies within the mini-max search in two-player games have been investigated extensively in the past. In [15, 16, 2, 3, 4, 5, 17], the primary cause of pathologies was deemed to be the independence of heuristic values of the leaf nodes. Such games were called non-incremental. Large branching factors were also considered contributing to a pathology. Later, [14] added non-inertness (i.e., a constant branching factor) to the list of suspects.

More recent work considered single-agent heuristic search and demonstrated that pathologies are possible even with admissible and consistent heuristic functions [6]. In
this paper we extend the previous efforts in the following ways: (i) several performance metrics (e.g., overall solution quality, total running time, etc.) are introduced for single-agent heuristic search and lookahead pathologies are shown for all of them; (ii) the demonstration is carried out in the standard testbed of the 8 puzzle as opposed to more contrived and artificial environments; (iii) finally we show that using meta-level control of the search via selecting the lookahead depth dynamically has a significant potential.

3 Lookahead Pathologies

Throughout the paper we will assume the standard RTA* real-time heuristic search model [11] as it is a general and modular technique allowing for further extensions (e.g., SRTA* and DTA* [22] or RTDP [1]). Thus, we consider a single-agent heuristic search in a discrete state domain with a finite number of deterministic actions. The states (set $S$) and actions (set $A$) form a directed graph with certain specified vertices representing the goal states. The edges (actions) are weighed with action costs: $c : A \rightarrow R$. The agent is provided with a perfect domain model: $\delta : S \times A \rightarrow S$.

We define the true distance-to-goal function $h^*(s)$ as the sum of action costs along the shortest path from state $s$ to the closest goal state. Generally speaking, the agent uses an approximation $h$ to the unavailable $h^*$. The approximation is typically inaccurate in some as: $\exists s \in S \left[h^*(s) \neq h(s)\right]$.

For a fixed starting state $s$, function $g(s')$ is defined as the sum of action costs along the shortest path from $s$ to $s'$. Finally, the sum of $h$ or $h^*$ and $g$ is typically denoted by $f$ or $f^*$. It is easy to see that $f^*$ remains constant along any optimal path from a fixed state $s$ to the closest goal. Also note that, for any state $s'$, action $a_1$ is inferior to action $a_2$ iff $f^*(\delta(s', a_1)) > f^*(\delta(s', a_2))$.

Located in the state $s$, the agent can use its perfect model $\delta$ to predict the states it will get to upon taking various sequences of actions. Two depth 2 lookahead trees are illustrated in Figure 1. RTA* defines policy $\pi(s, p)$ as follows: (i) consider $s$’s immediate children $\{c_i\}$; (ii) for each child $c_i$ construct the lookahead search trees of $p$ plies deep by envisioning terminal states of all action sequences of $p$ actions (whenever possible); (iii) evaluate the leaf nodes of each lookahead tree rooted in $c_i$ using the $f$ function and select the minimum-valued state which becomes the lookahead-augmented value of the $c_i$; (iv) output the single action leading to the child $c_i$ with the minimum lookahead-augmented $f$-value (resolution ties randomly).

Additionally, a hash table of all previously visited states is maintained. The $f$-value of a state $x$ is drawn from the table whenever $x$ has been previously visited. After each move, the hash-table value of the state $s$ just left is updated with the second-best $f$-value among its children $c_i$. This mechanism prevents infinite looping in RTA* under certain reasonable conditions [11].

Depending on the lookahead depth, random tie resolution, and the approximate heuristic $f$, the action $a$ output by $\pi(s, p)$ can be suboptimal: $\exists a^* \neq a \left[f^*(\delta(s, a^*)) < f^*(\delta(s, a))\right]$. The probability of such an error for state $s \in S$ is denoted by $Err(s, p)$. Additionally, we will consider the expected value of $Err(s, p)$ over the states $s$. Such state-independent quantity will be referred to as $Err(p)$.

Suboptimal actions lead to suboptimal solutions (i.e., paths from a starting state $s_0$ to a goal state longer than $h^*(s_0)$). If the ply level varies in a fixed range $[0, p_{\text{max}}]$ the solution length can vary as well. Ply $p^*$ is optimal for starting state $s_i$ if the solution produced by RTA* with ply depth $p^*$ starting in state $s_i$ is the shortest.

Finally, we consider the total number of states generated by RTA* while it searches for a solution. This metric depends on the total number of moves as well as the number of states generated for each move. We are now ready to define several types of pathologies as follows:

Type 1 pathology. Deeper lookahead is pathological iff RTA* is expected to find worse (i.e., longer) solutions.

Type 2 pathology. Looking deeper ahead at each node leads to more nodes being generated (often exponentially more) usually resulting in longer total search time. Thus, looking deeper ahead is pathological iff RTA* is expected to generate fewer nodes in total.

Type 3 pathology. Deeper lookahead search is pathological iff looking deeper ahead is expected to increase the chances of taking a suboptimal action in a state.

![Figure 1. RTA* with the lookahead depth of 2 in action.](image-url)
4 Empirical Evidence

The 8 sliding tile puzzle is a common testbed used in the field of heuristic search [11, 21] as the general $N \times N - 1$ extension of the puzzle is NP-hard [20]. In this study a single 8-puzzle problem instance is defined by its starting state which is drawn randomly from the total space of $9! / 2 = 181440$ solvable states. RTA* is run on 1,000 random problem instances using various heuristic functions and lookahead depths. All three pathology types have manifested themselves.

Type 1 pathology can be observed with a heuristic function represented by an Artificial Neural Network. We used an 81-100-31 single-hidden layer feed forward network to approximate the true $h^*$ value. The network was trained offline and then used in the RTA* search. The best expected solution length of 68 is achieved with the ply depth of 5 and degrades to 115 moves when the lookahead of depth 10 is used (Figure 2). Likewise, the “Unif” heuristic with the values uniformly distributed in $[0, h^*(s)]$ for every state $s$ is pathological as shortest solutions are expected with the lookahead depth of 1.

Type 2 pathology can be observed with heuristic functions represented by an Artificial Neural Network, a decision tree*, and a k nearest neighbors classifier. All of them were trained off-line to approximate the true $h^*$ value. As Figure 3 demonstrates, increasing the lookahead depth actually decreases the expected overall number of nodes generated by RTA* and, consequently, the total time it spends on an average problem instance.

Type 3 pathology is seen with the Artificial Neural Network as well the uniformly distributed heuristics. In both cases, the chances of RTA* taking a suboptimal action (i.e., moving to a state which is not on an optimal solution path from the current state) increase as the lookahead depth increases (Figure 4).

5 Toward Efficient Search Control

Real-time heuristic search algorithms such as RTA* trade the optimality of the solution for faster problem-solving times. For instance, RTA* can solve large instances of the $N \times N - 1$ puzzle (e.g., the 99-puzzle) in a reasonable amount of time [11] while applying complete searches to even the 24-puzzle requires carefully crafted domain-specific heuristic functions and substantial running time [12].

We are interested in automatic construction of problem-solvers. Numerous previous research has focused on automatic discovery of heuristic functions via constraint relaxation [18], limiting the subgoal interactions [12], abstraction [19], and pattern databases [7]. In the field of reinforcement learning heuristics (i.e., value functions) can be machine learned on- and off-line (e.g., [1]).

Unfortunately, as this paper and [6] demonstrate, having a reasonable heuristic is not sufficient to ensure the efficacy of the search. Consider Table 1 where the first col-

---

*I should be noted that the decision tree represented heuristic is the best of the 7 heuristic functions studied. While being inadmissible and inconsistent, it outperforms the Manhattan distance in terms of the RTA* solution length as well as the total number of nodes generated.
unn lists various heuristics used with RTA* in the 8-puzzle: “ANN” stands for Artificial Neural Network, “Unif” is the uniformly distributed heuristic, “Tree” is the decision tree, “NM” is the Hamming distance, “0” is the zero-everywhere heuristic, “kNN” is the k nearest neighbor, and “MD” is the Manhattan Distance. The table shows the expected RTA* solution lengths for different lookahead depths and different heuristics. There are two important observations to make:

1. due to type 1 pathologies, selecting a static lookahead depth properly would sometimes considerably increase the quality of the solutions (e.g., RTA* with ANN-represented heuristic at depth 5 spends 27 times less time while finding 1.7 times better solutions);

2. selecting lookahead depth dynamically for each starting state has an even more impressive potential. For instance, in RTA* guided by the Manhattan distance the expected solution length can be reduced from 35 to 26 and at the same time the expected solution time can be decreased.

Thus, an important parameter in real-time heuristic search is the depth of lookahead. While this fact has been generally recognized in the two-player game community (cf. search extensions for alpha-beta search), the corresponding efforts in real-time single-agent heuristic search have been more limited (e.g., DTA* in [22]).

The objective of our on-going research is to design a practically efficient and theoretically well-founded metalevel control module to guide the search within an RTA* -like algorithm thereby increasing the overall efficacy and applicability of real-time decision-making. This paper takes a step toward this goal.

Acknowledgments

Valeriy Bulitko, Russell Greiner, Robert Holte, Richard Korf, Sven Koenig, Lihong Li, and Ken Wong have contributed in various ways. The research has been supported by the University of Alberta, NSERC, and AICML.

References


